

Intel Array Visualizer

HDF Workshop VI

December 5, 2002

John Readey

john.readey@intel.com

Introduction

- Intel Array Visualizer is a software tool for data visualization included with Intel Fortran for Windows v7.0
- Derived from Compaq's Array Visualizer
- Includes
 - Array Viewer: Viewing application
 - Library routines: API for C and Fortran applications
 - Object Model: COM based class library
 - ActiveX Controls: Re-usable UI components

Array Viewer

Intel(R) Array Viewer - tall.h5

File Edit View Go Help

Hide Locate Previous Next Back Forward Stop Refresh Start Print

Edit Path /g1/g1.1/dset1.1.1

Root

- g1
 - dset1.1.1
 - attr1
 - attr2
 - dset1.1.2
 - g1.2
 - g1.2.1
 - slink
- g2
 - dset2.1
 - dset2.2
 - attr1
 - attr2

Integer*4 dset1.1.1(10, 10)

Hide Data Class: SDS

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	
2	0	2	4	6	8	10	1
3	0	3	6	9	12	15	1
4	0	4	8	12	16	20	2
5	0	5	10	15	20	25	3
6	0	6	12	18	24	30	3
7	0	7	14	21	28	35	4
8	0	8	16	24	32	40	4
9	0	9	18	27	36	45	5

dset1.1.1(0, 0) = 0

Show Dimension Info

Hide Attributes

Attributes:

Name	Size	Type
attr1	135 B	Attribute
attr2	135 B	Attribute

2 objects

Array Viewer - Features

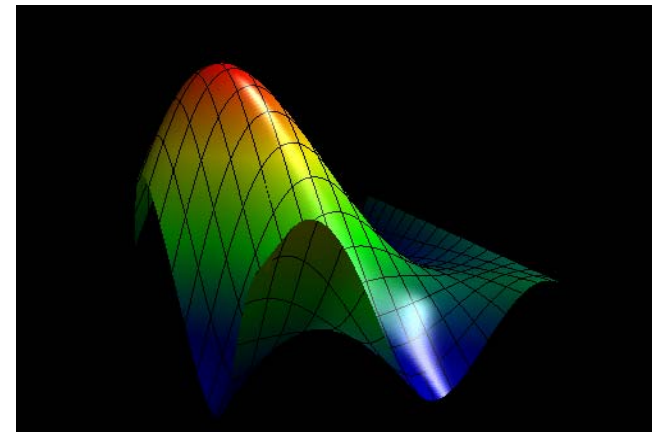
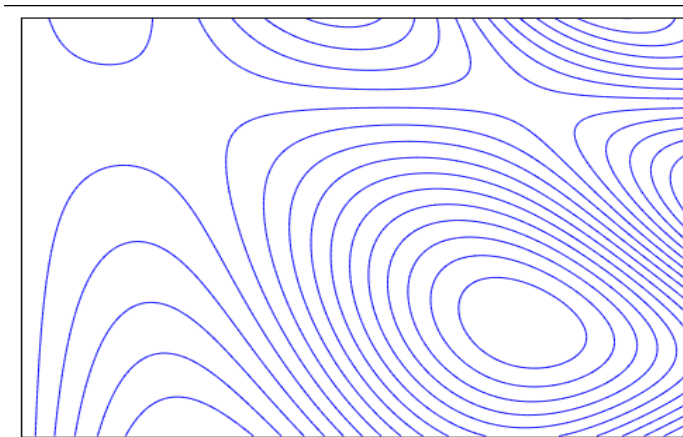
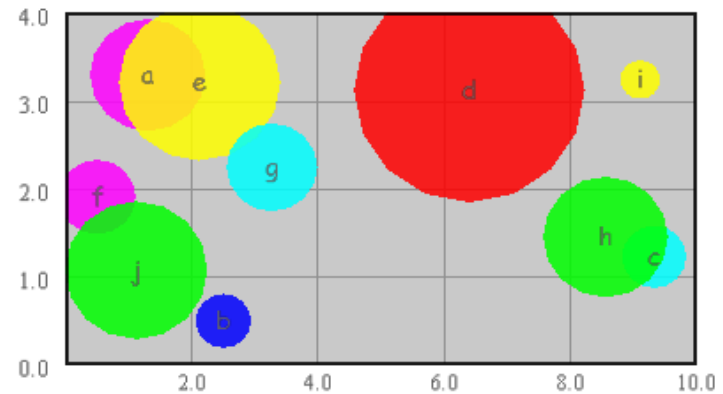
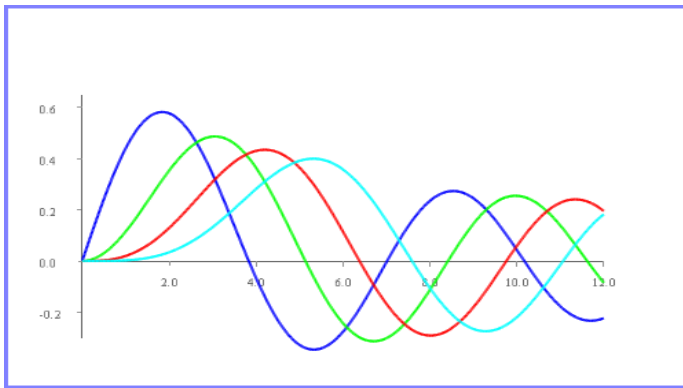
- Data visualization program for viewing HDF4, HDF5 files
 - Also supports XML, BMP, GIF, JPG, PNG
 - Incremental Load/Save (for HDF4, HDF5)
 - HDF4 support didn't make it into the 7.0 product
 - Will be available in the 7.0.1 release (Q2 '03)
- Browser-like interface
- Edit/View mode
- Data Grid for displaying Datasets, Attributes

Array Viewer – Tree Pane

- Groups, Datasets, Attributes, and Links displayed as icons in a tree control
- Clicking on icon displays object in right pane
- Graphs and Pages also displayed in the tree
 - Graph: a collection of plots, axes, and captions
 - Page: HTML/script code
- Copy/Paste, Drag&Drop supported

Viewer Visualization Features

- Variety of 2D/3D plot types: Image, XY, Contour, Heightmap, Vector, Log plots



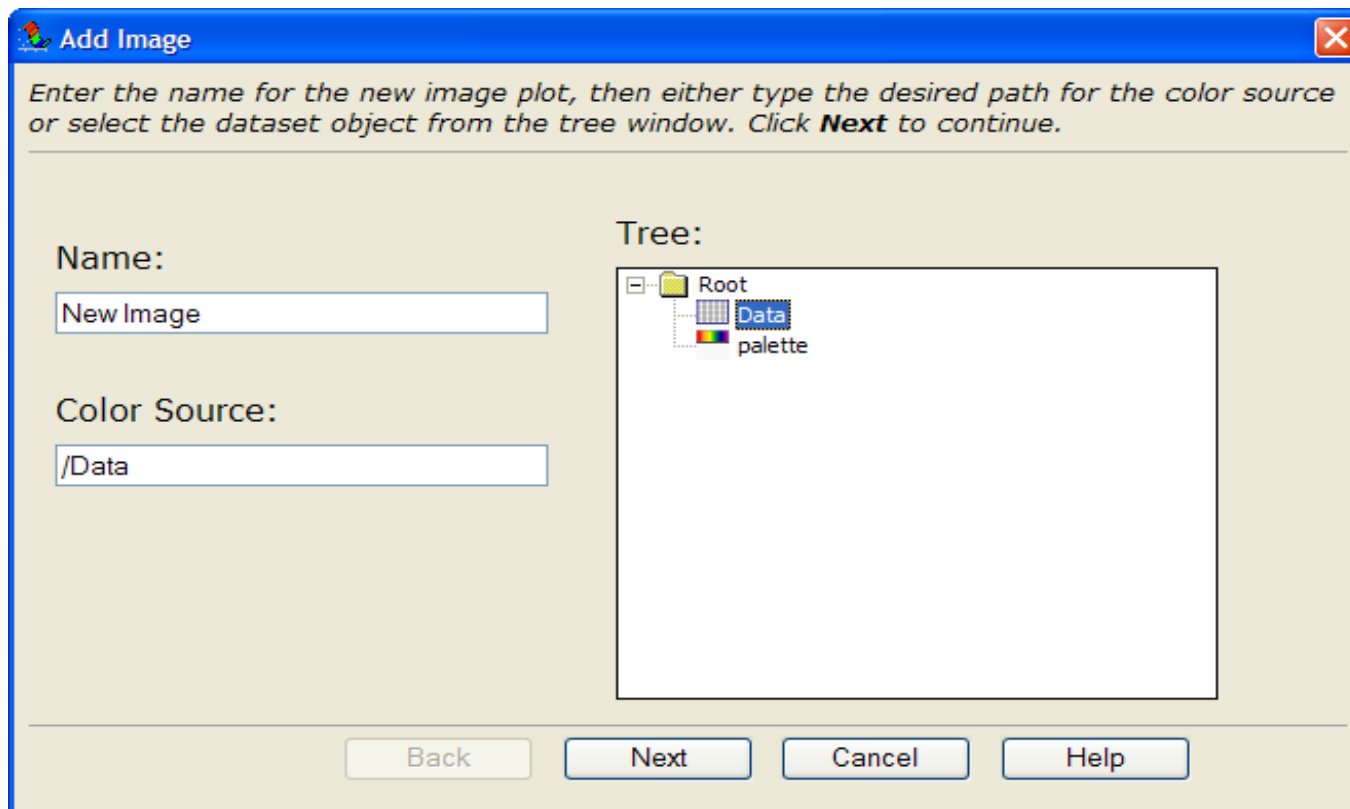
Viewer Visualization Features - Cont

- Images can be indexed or True Color
- Color mapping functionality based on HDF5 Image and Palette Specification
- Multiple Images can be composited



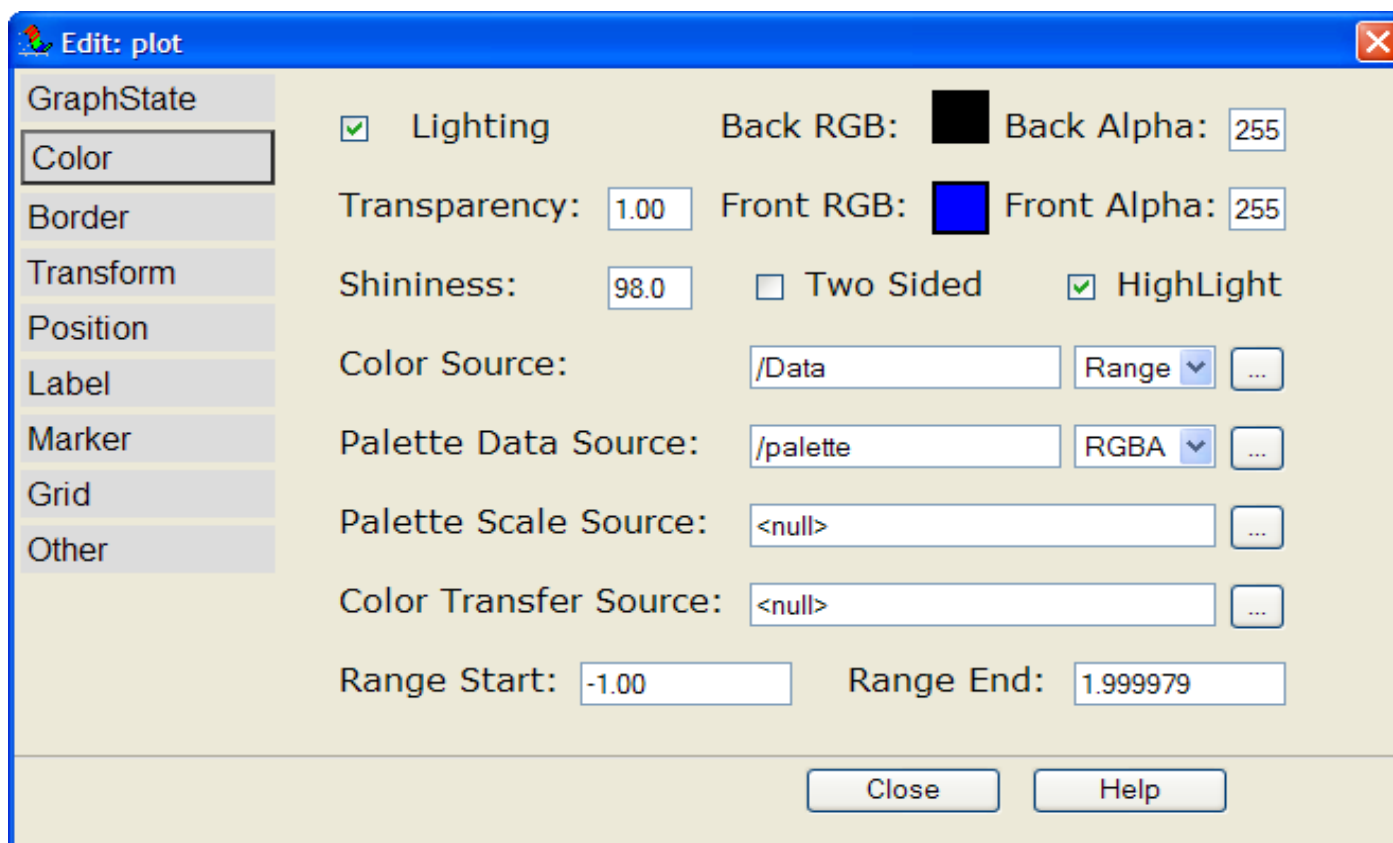
Viewer Visualization Features - Cont

- Wizards provided for the creation of new plots



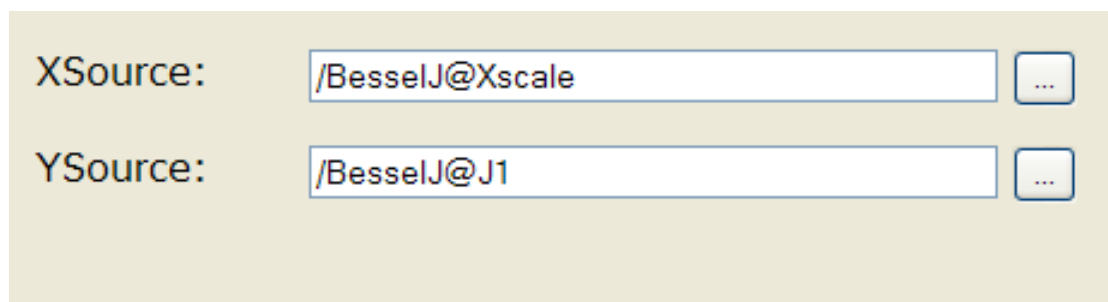
Viewer Visualization Features - Cont

- Property pages enable plot appearance to be modified



Viewer Visualization Features - Cont

- Graphs are collections of plots, axes, and captions
- Data for plots or axes is referenced as a path to a dataset
- Paths can contain suffix to indicate section and/or sub-type



A screenshot of a configuration window with a light beige background. It contains two rows of input fields. The first row is labeled 'XSource:' and has a text box containing '/BesselJ@Xscale' followed by a small square button with three dots. The second row is labeled 'YSource:' and has a text box containing '/BesselJ@J1' followed by a similar button with three dots.

Viewer – Page Objects

- Page objects are HTML code that can contain:
 - Standard HTML elements
 - Graph and grid objects
 - UI elements (buttons, text entry, checkboxes, etc)
 - Script code for dynamic behavior
- Used to:
 - Group related datasets, graphs, explanatory text in one view
 - Created interactive views
- Page data is saved to the file along with other elements (in HDF4/HDF5 as a group attribute)

Viewer – Page Example 1

The screenshot shows the Intel(R) Array Viewer application window titled "Intel(R) Array Viewer - tall_wPages.xml". The interface includes a menu bar (File, Edit, View, Go, Help), a toolbar with icons for Hide, Locate, Previous, Next, Back, Forward, Stop, Refresh, Start, and Print, and a path bar showing "page:/Show datasets".

On the left, a tree view displays the file structure:

- Root
 - g1
 - g2
 - Show datasets
 - attr1
 - attr2

The main content area displays the text: "This page displays !TWO! datasets using the grid control."

Dataset #1: /g2/dset2.1

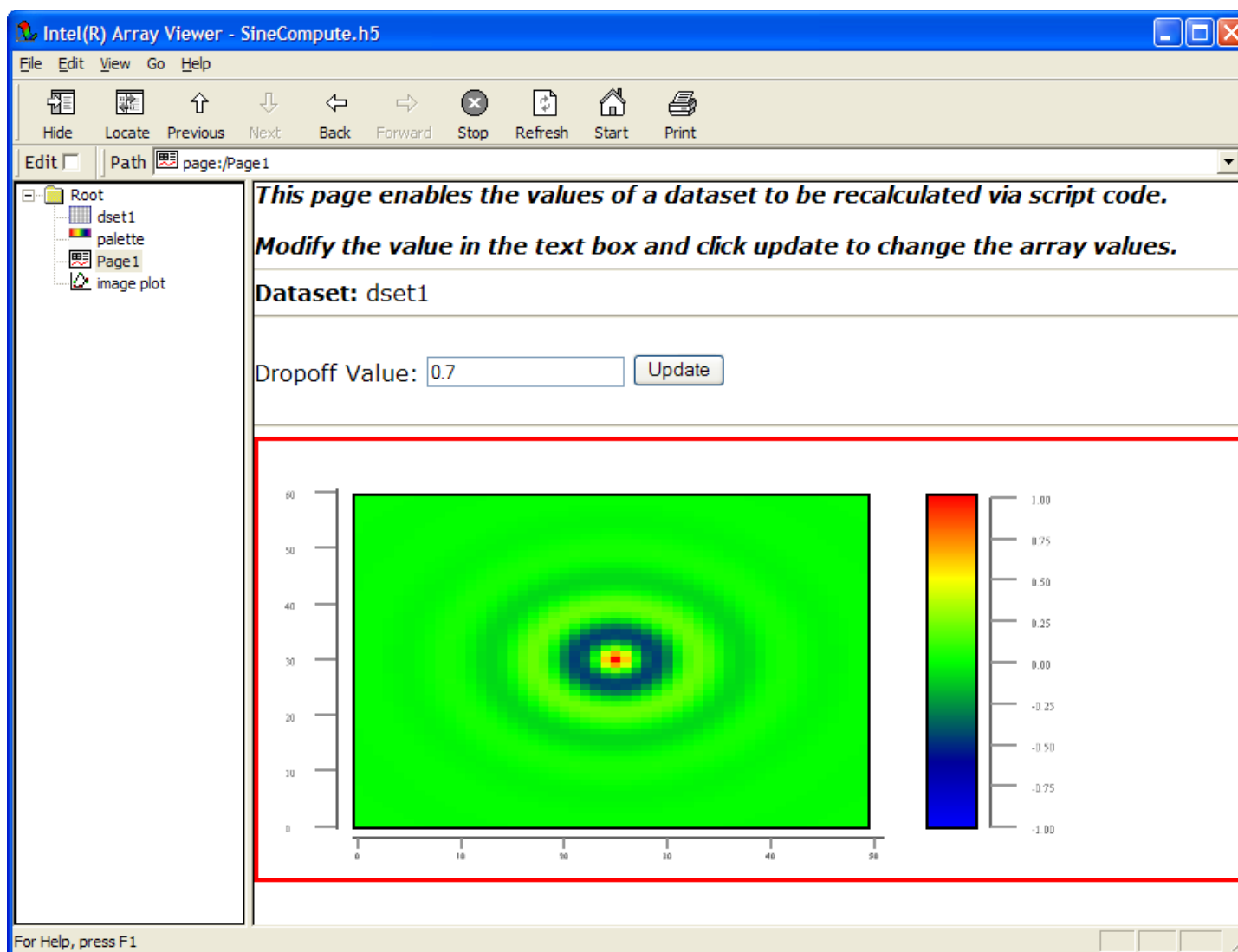
0	1.000000
1	1.100000
2	1.200000
3	1.300000
4	1.400000
5	1.500000
6	1.600000
7	1.700000
8	1.800000
9	1.900000

Dataset #2: /g2/dset2.2

	0	1	2	3	4
0	0.000000	0.100000	0.200000	0.300000	0.400000
1	0.000000	0.200000	0.400000	0.600000	0.800000
2	0.000000	0.300000	0.600000	0.900000	1.200000

At the bottom left, it says "5 objects".

Viewer – Page Example 2



Library Routines

- Provides means for C/Fortran programs to read and write data to a file (HDF4, HDF5, or XML)
- Only avOpen, avSave calls access files directly
- Other File I/O is implicit
- For HDF4, HDF5 files:
 - Datasets, Groups loaded from file as they are accessed
 - avSave writes dirty objects back to file
- For XML files:
 - All objects are loaded on avOpen
 - avSave rewrites the entire file
 - Loops are replace by links

Library Routines - Cont

- File read example in C:

```
// open file
avOpen("tall.h5", FALSE);
// get extent of dimension 1
nExtent = avGetExtent("/g1/g1.1/dset1.1.2", 1);
// get pointer to dataset
pData = (long *)avGetArrayPointer("/g1/g1.1/dset1.1.2");
// Print out the contents of the array
printf("array elements:\n");
for (i=0; i<nExtent; i++) {
    printf("pData[%d]: %d\n", i, pData[i]);
}
```

Library Routines - Cont

- File save example in Fortran:

```
integer, parameter :: numcols=4, numrows=5
real(4) :: M(numcols, numrows)
integer :: dim2d(2)
! Initialize array data
do j=1, numrows
  do i=1, numcols
    M(i, j) = i*0.01+j
  end do
end do
! Add M to the "watch list"
dim2d = shape(M)
call avStartWatch(LOC(M), 2, dim2d, AV_REAL4, "M", status)
! Save all arrays in watch list to HDF5 file
call avSave("mydata.h5", status)
```

Library Routines - Cont

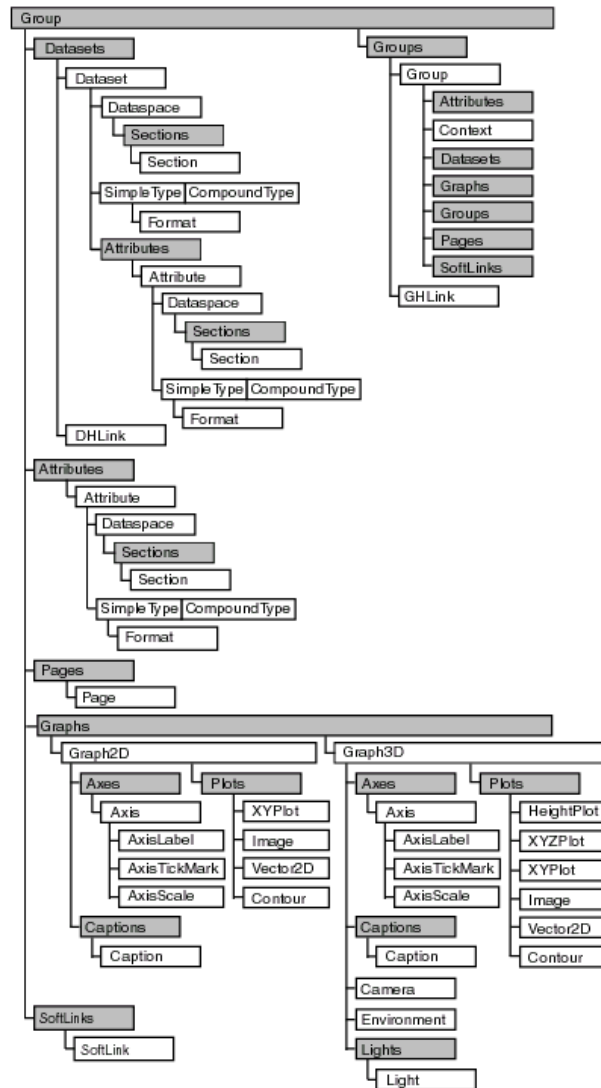
- avNewViewer function can be used to invoke the Array Viewer
- Viewer example in Fortran:

```
integer, parameter :: numcols=4, numrows=5
real(4) :: M(numcols, numrows)
integer :: dim2d(2)
! Initialize array data
do j=1, numrows
    do i=1, numcols
        M(i, j) = i*0.01+j
    end do
end do
! Add M to the "watch list"
dim2d = shape(M)
call avStartWatch(LOC(M), 2, dim2d, AV_REAL4, "M", status)
! Create a viewer instance
call avNewViewer(viewerId)
! Make it visible
call avVisible(viewerId, 1, status)
```

Object Model

- COM based class library
- 40+ classes representing datasets, dataspace, types, groups, links, graphs, plots, etc.
- Provides more fine-grained control than C/Fortran lib (but not direct file access)
- Organized in hierarchy:
 - Class properties link to sub-objects
 - Example: mydataset.Dataspace
 - Collection classes contain an arbitrary number of objects of a given type
 - Example: mygroup.Groups[“mysubgroup”]

Object Model Diagram



Object Model – Language Support

- C++, Fortran:
 - best performance 😊
 - somewhat tedious to program ☹️
- .Net languages (C#, VB.Net):
 - not as efficient as C++/Fortran, better than script
 - easy to program (+ Intellisense) 😊
- Script (JavaScript, VBScript)
 - not very efficient (but often good enough) ☹️
 - easy to program 😊
 - no debugger ☹️
 - can use code in Page objects 😊

List Datasets Example – C++

```
//  
// Print name and rank of all datasets in the group  
//  
void ListDatasets(IAvGroup *pGroup)  
{  
    long nCount;  
    IAvDatasets *pDatasets;  
    pGroup->get_Datasets(&pDatasets);  
    pDatasets->get_Count(&nCount);  
    for (int i=0; i<nCount; i++)  
    {  
        IAvDataset * pItem;  
        VARIANT var;  
        VariantInit(&var);  
        var.vt = VT_I4;  
        var.lVal = i+1; // 1-base index  
        pDatasets->Item(var, &pItem);  
        // Get the name of the dataset  
        BSTR bstrName;  
        pItem->get_Name(&bstrName);  
        // Get the rank  
        long nRank;  
        IAvDataspace pDataspace;  
        pItem->get_Dataspace(&pDataspace);  
        pDataspace->get_Rank(&nRank);  
        pDataspace->Release(); // release dataspace  
        pItem->Release(); // release dataset  
        wprintf(L"name: %s rank: %d\n", bstrName, nRank);  
        SysFreeString(bstrName); // free the string  
    }  
}
```

List Datasets Example – C#

```
//  
// Print name and rank of all datasets in the group  
//  
void ListDatasets(Avis.Objmod.Group group)  
{  
    foreach (Avis.Objmod.Dataset item in group.Datasets)  
    {  
        string name = item.Name;  
        int rank = item.Dataspace.Rank;  
        Console.WriteLine("name: {0} rank: {1}", name, rank);  
    }  
}
```

List Datasets Example – JavaScript

```
//  
// Print name and rank of all datasets in the group  
//  
function ListDatasets(group)  
{  
    for (var i=0; i<group.Datasets.Count; i++)  
    {  
        var dset = group.Datasets.Item(i+1);  
        var name = dset.Name;  
        var rank = dset.Dataspace.Rank;  
        WScript.Echo("name: " + name + " rank: " + rank);  
    }  
}
```

Creating Datasets Example

JavaScript

```
// Create a dataset that will hold an array of strings
var aStringArray = new Array("AIRS", "CERES", "MODIS", "AMSU", "AMSR-E", "HSB");
var dset1 = root.Datasets.CreateDataset("dset1"); // new dataset
dset1.WriteData(aStringArray); // type and space set implicitly

// Create a dataset that will hold an array of longs
var aIntArray = new Array(2, 3, 5, 7, 11, 13); // javascript array
var tLong = new ActiveXObject("Avis.AvSimpleType"); // av type object
tLong.TypeID = 5; // signed 4-byte integer
var sSpace = new ActiveXObject("Avis.AvDataspace"); // dataspace object
sSpace.Rank = 1;
sSpace.Extent(1) = aIntArray.length; // set dimension extent
var dset2 = root.Datasets.CreateDataset("dset2"); // new dataset
dset2.Allocate(tLong, sSpace); // allocate memory for the dataset
for (var i=0; i<aIntArray.length; i++)
{
    dset2.WriteElement(i, aIntArray[i]); // write the dataset element
}
```

Read Element Example

JavaScript

```
var fileLoader = new ActiveXObject("Avis.AvFileLoader");
var datafile = GetCurDirectory() + "/tall.h5";
WScript.Echo("Opening: " + datafile);
fileLoader.Load(datafile);
var root = fileLoader.Root;
var g1Grp = root.Groups("g1");
var g11Grp = g1Grp.Groups("g1.1");
var dset112 = g11Grp.Datasets("dset1.1.2");
// Read dset1.1.1 element by element
var dset111 = g11Grp.Datasets("dset1.1.1");
var indx = new Array(2);
for (var i=0; i<dset111.Dataspace.Extent(2); i++) {
    indx[1] = i;
    for (var j=0; j<dset111.Dataspace.Extent(1); j++) {
        indx[0] = j;
        var elmt = dset111.ReadElement(indx);
        WScript.Echo("dset1.1.1[" + i + "][" + j + "]: " + elmt);
    }
}
// Read dset1.1.2 to javascript array
var vbArray = new VBArray(dset112.ReadData());
var retArray = vbArray.toArray();
for (var i=0; i<retArray.length; i++) {
    WScript.Echo("dset1.1.2[" + i + "]: " + retArray[i]);
}
```

Reading Compound Elements

- For datasets of compound types ReadElement() returns an object
- Properties of the object are the fields of the type
- Fields that have an extent > 1 become indexed properties
- Fields that are themselves compound types become sub-objects of the returned object

Compound Elements Example

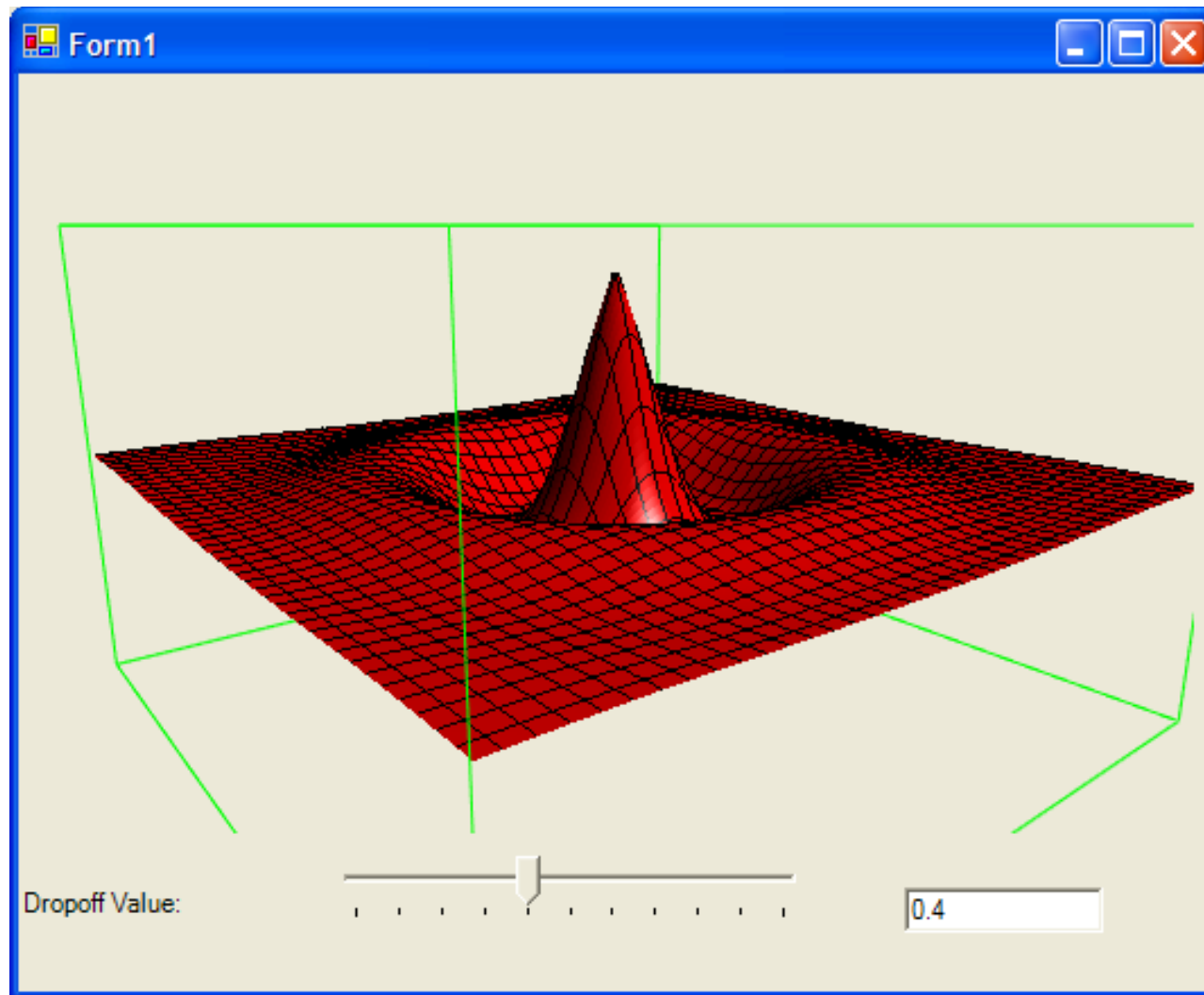
JavaScript

```
var fileLoader = new ActiveXObject("Avis.AvFileLoader");
var datafile = GetCurDirectory() + "/tcompound.h5";
WScript.Echo("Opening: " + datafile);
fileLoader.Load(datafile);
var root = fileLoader.Root;
var dset1 = root.Datasets("dset1");
// Read dset1
for (var i=0; i<dset1.Dataspace.Extent(1); i++) {
    var elmt = dset1.ReadElement(i);
    var a = elmt.a_name;
    var b = elmt.b_name;
    var c = elmt.c_name;
    WScript.Echo("dset1[" + i + "]: (" + a + ", " + b + ", " + c + ")");
}
// Read dset3
var group1 = root.Groups("group1");
var dset3 = group1.Datasets("dset3");
for (var i=0; i<dset3.Dataspace.Extent(1); i++) {
    var elmt = dset3.ReadElement(i);
    var x = elmt.int_array(1);
    var y = elmt.float_array(1).float_array-Sub1(1);
    WScript.Echo("dset3[" + i + "]: (" + x + ", " + y + ")");
}
```

ActiveX Controls

- User Interface components that can be used to create GUI applications
- Controls supported in Visual C++, Visual Basic, Compaq Visual Fortran
- Graph, Grid, Tree controls supply most of the functionality in Array Viewer
- Each control has a limited number of properties
 - Most state is accessed through object model
- Events signal changes of state

ActiveX Control Example



File Loaders

- Each file format supported by Array Visualizer is implemented by a separate file loader component
- File loaders run in their own address space
- Additional file formats can be supported by registering a new file loader on the system
- No source changes, re-linking required for applications
- Instructions for writing file loaders not documented this release
- Must be written in C++ (Fortran support planned)

Getting the Software

- Go to <http://www.intel.com/software/products>
- Order or download Fortran for Windows v7.0
- Free evaluation available
- Post questions or comments on Fortran forum
 - Click “User Forums” in the above web page
- Let us know what features you’d like to see in future versions